Estimating Power Consumption of Containers and Virtual Machines in Data Centers

Xusheng Zhang, Ziyu Shen, Bin Xia, Zheng Liu, Yun Li*
Jiangsu Key Laboratory of Big Data Security & Intelligent Processing
Nanjing University of Posts and Telecommunications
liyun@njupt.edu.cn

***Abstract*—Virtualization technologies provide solutions of cloud computing. Virtual resource scheduling is a crucial task in data centers, and the power consumption of virtual resources is a critical foundation of virtualization scheduling. Containers are the smallest unit of virtual resource scheduling and migration. Although many effective models for estimating power consumption of virtual machines (VM) have been proposed, few power estimation models of containers have been put forth. In this paper, we offer a fast-training piecewise regression model based on decision tree to build a VM power estimation model and estimate the containers' power by treating the container as a group of processes on the VM. In our model, we characterize the nonlinear relationship between power and features and realize the effective estimation of the containers on the VM. We evaluate the proposed model on 13 workloads in PARSEC and compare it with several models. The experimental results prove the effectiveness of our proposed model on most workloads. Moreover, the estimated power of the containers is in line with expectations.**

***Index Terms*—power estimation, containers, virtual machines, data centers**

I. Introduction

The fast-growing power consumption of computational infrastructures takes up a significat portion of the overall building and maintenance cost of data centers. Accordingly, many virtual resource scheduling approaches incorporated power consumption into the performance metrics [1]–[3]. Accurate estimation and forecast of power consumption in data centers is the cornerstone of better virtual resource scheduling and efficient power management, resulting in reduced electricity bills. The power estimation of containers is meaningful because containers are the smallest unit of virtual resource scheduling and migration in data centers, whether the container is running on the physical server or running on the Virtual Machine (VM). Many solutions of VM power estimation have been developed [4]–[10], but few works focus on the power estimation of containers [11]–[13]. To the best of our knowledge, there is no container power estimation model considering the scene where the containers are running on VMs.

This work was supported by National Key Research and Development Program of China under grant 2018YFB1003702 and Graduate Student Scientific Research Innovation Program of Jiangsu Province under grant KYCX20_0760.

*Corresponding author: Yun Li

With the above observation, we are motivated to develop a solution to container power estimation in the scene of containers running on VMs. The difficulty of the power estimation of containers running on VMs is the Performance Monitoring Counters (PMC) commonly used in VM power modeling is unavailable in such environments. Therefore, we select Operating System Metrics (OSM) as features instead. Stemming from the demand of real-time container scheduling in data centers and the problem of the nonlinear relationship between VM features and power in some servers, we adopt a fast-training piecewise regression model based on decision tree. We replace each leaf node of the decision tree with a linear model to realize the further estimation of containers. In order to increase model flexibility and reduce model complexity, we propose a universal model suitable for different VM types by summing the feature vector of the same type of VMs together. Regarding a container as a group of processes running on the VM, containers' power can be estimated by the VM model when the features of containers can correspond to the features of VMs one to one. After validating our model on the workloads in benchmarks, the results show that our model outperforms the other models on most workloads, and the containers' power is estimated as expected.

The rest of the paper is organized as follows. In Section II, we review the related works on power estimation of VMs and containers. In Section III, we propose a universal power estimation model for VMs and containers and elucidate its feasibility. We introduce our experimental setup and evaluate the performance of the model in Section IV. Finally, in Section V, we summarize this paper.

II. Related Works

There are mainly two kinds of power estimation models for VMs and containers. The one is regression model. The other is decompostion model. For VM power estimation, numerous variants of linear regression were used [7]–[10]. Yu et al. adopted Long Short Term Memory (LSTM) [14] to find the correlation between the features and used Attention [15] to improve regression generalization [4]. Jiang et al. decomposed the physical machine's power to each VM's power by non-deterministic Shapley Value to realize the fair allocation [5].

For container power estimation, the primary idea behind regression models is also linear regression [12], [13]. Brondolin

978-1-7281-6677-3/20/$31.00 ©2020 IEEE
DOI 10.1109/CLUSTER49012.2020.00039

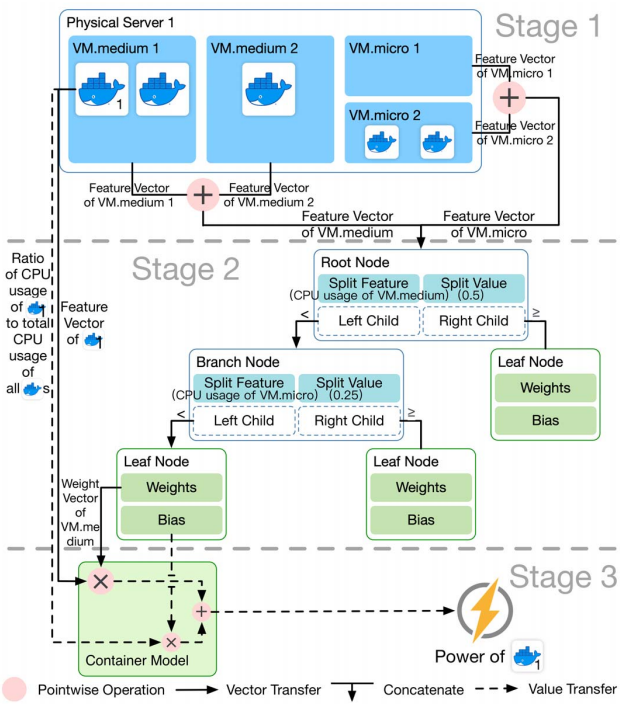

Fig. 1. The framework of the proposed model.

et al. decomposed the Running Average Power Limit (RAPL) [16] value of the CPU cores to the power of each thread [11]. Subsequently, the power of all threads run by the container was added up as the container power. Although Piraghaj et al. calculated the server power by the CPU utilization of containers on VMs, they did not focus on the container power estimation [2].

III. Power Estimation of Containers and VMs

Fig. 1 is the overall framework of our model. First, we elaborate the principle of VM power estimation using the decision tree in Section III-A. After the decision tree in Fig. 1 has been trained, we use it for further power estimation. When new data arrive, we sum the feature vector of the same type of VMs (e.g., VM.medium in Fig. 1) and concatenate the feature vectors of all VM types in stage 1, as explained in Section III-A. We can find the corresponding leaf node in the light of the split conditions and record the leaf node's model parameters in stage 2. After that, models corresponding to each VM (type) can be accessed. The container's power can be estimated by substituting the features of the container running on the VM for the corresponding features of the VM type in the model. The container power estimation process of stage 3 is put forward in Section III-B.

A. Type-based VM Power Estimation

In a data center, the power of physical servers can be measured by Power Distribution Units (PDU) or power meters, but the power of VMs and containers cannot. Our model's main idea is establishing a relationship between the physical server power and the relevant features of the VMs running on the server and then applying the model to the containers running on the VMs.

In physical servers, the leading cause of power consumption is the operation and management of VMs. When there is no VM running, the server will also consume power, which is called idle power P_{idle}. Therefore, the power of the physical server P_{ser} is equal to P_{idle} plus the dynamic power P_{dyn} (the total power generated by all VMs on the server).

However, we cannot simply model each VM due to the dynamic scheduling of VMs on the physical servers. Retraining when new VMs come or old VMs leave would result in more power consumption and longer waiting time of the power estimation, contrary to the original purpose of saving power.

By looking up the products providing by Cloud Service Providers (CSP)[1], we notice that VM types (or called VM configurations) are usually fixed. For the same type of VMs on the same server, the power of VMs is the same in case the feature vector of the VMs is the same.

For each VM on the server, we assume that the acquired VM feature vector of the j-*th* VM belonging to the i-*th* VM type is X_{ij}. According to the paper [9], we can establish a linear model between the VM's features and power. Based on the above discussion, P_{dyn} can be expressed as follows,

$$\begin{aligned} P_{dyn} &= \sum_{i}^{m}\sum_{j}^{n_i} P_{VM_{ij}} = \sum_{i}^{m}\sum_{j}^{n_i} \left(W_i^T X_{ij} + b_i\right) \\ &= \sum_{i}^{m} W_i^T \sum_{j}^{n_i} X_{ij} + B = \sum_{i}^{m} W_i^T X_i^{sum} + B \\ &= W_{all}^T X_{all} + B, \end{aligned} \tag{1}$$

where m is the number of VM types on the server, and n_i is the number of VMs belonging to each VM type. W_i and b_i represent the weight vector and the bias term of the i-*th* VM type, respectively. B is the sum of all VMs' bias terms. X_i^{sum} is the sum of feature vectors of all VMs belonging to the i-*th* VM type. $W_{all} = [W_1, \cdots, W_m]$ and $X_{all} = [X_1^{sum}, \cdots, X_m^{sum}]$ represent the concatenation of all VM types' weight vectors and the concatenation of all VM types' feature vectors, respectively.

According to (1), X_i^{sum} means we can regard the feature vectors' sum of the same type of VMs as a single feature vector to train the model. To ensure the accuracy of the following container power estimation, the weights in W_i and the bias b_i should be non-negative. When the weight value of one feature is negative, it means the component's operation decreases power. Nevertheless, it is not realistic. Besides, the meaning of the bias is the idle power of the VM (When all values in X_{ij} are zero), so the bias b_i should be non-negative.

In order to make servers more power-efficient, many hardware manufacturers have introduced power-saving technolo-

[1]https://aws.amazon.com/ec2/instance-types/

gies, such as Intel SpeedStep [17], an implementation of Dynamic Voltage Frequency Scaling (DVFS) [18]. The appearance of these technologies makes the relationship between server power and features nonlinear, which degrades original linear model performance.

We observe that as the number of full-load physical cores (Intel Core i5-6500) with Intel SpeedStep enabled increases, the increment of power consumption becomes smaller. The power of the server increases 21W from the idle state to the single-core full-load state, while the power of the server increases about 10W for the next several cores. The foremost cause of this phenomenon is, in the idle state, the server will sleep or shut down some components (such as the LLC, core clock, etc.) to save power. As a result of the recovery from the idle state to the working state, these components need to be reenergized, resulting in a nonlinear change of power. However, these component features causing nonlinearity are difficult to obtain, and the features are different under different architectures.

Considering that this phenomenon was also observed from Intel Xeon E5-2609, we decided to employ a piecewise linear model to solve the nonlinear relationship between power and features. Inspired by the paper [19], we adopt the decision tree for more accurate splitting.

The structure of our model is similar to the regression tree, but each leaf node stores W_{all} and B of (1). The structure of this tree is shown in stage 2 of Fig. 1. The information stored on the branch node is the split feature, the split point and two children. When the data divided into a leaf node, a linear model is trained and evaluated.

We adopt two pre-pruning strategies to avoid overfitting and accelerate the training process. The first is the error decrease threshold τ. For each feature c, we sort all distinct values from small to large and select the bisection point of two adjacent values as the optional split point t. For each optional split point t, we evaluate the test errors of two children's models by training and testing the linear models under the two subsets split by the split point and summimg the errors of two children up. If the decrease of error is more than the threshold τ after splitting, then this pair of split feature and the split point (c, t) will be an alternative. After all split points of all features are traversed, we consider the pair of split feature and split point (c_{best}, t_{best}) that has the smallest error as the best option. If none of the (c, t) can reduce the error beyond the threshold τ, then the node will not be divided and will be set as a leaf node to train the model. The second pre-pruning strategy is the minimum subset size γ. When models on leaves are trained in too small data sets, it is not conducive for the linear model to learn the real data distribution. Consequently, we stipulate that when the size of the divided subsets is less than the threshold γ, we will not add this split pair into the alternatives.

One of the problems with the selection of split points is that it will take much time to traverse these split points when there are many features, and each feature has a lot of distinct values. It will result in slow tree generation, which is contrary to our intention of fast response to provide power information for scheduling. Our solution to realize fast training is rounding the normalized values of split points to a specific floating-point precision (e.g., 10.2345's floating-point precision is 4) while ensuring the model's accuracy which is illustrated in Section IV-D. Parallel computing can also be adopted for a shorter training time.

B. Container Power Estimation

Considering that a container can be regarded as a group of processes running on the host machine [20], two prerequisites need to be satisfied for directly using the VM power model for containers running on the VM. The first is that the acquired container features can correspond to the features of the VM one to one. The other one is that the sum of containers' feature value should not exceed the VM's feature value at the same moment. Due to the interdependence between features [10], the bias term also represents the power consumed by containers. Considering the power contribution of CPU usage is much higher than the other features [19], we break the bias sum B down into the bias of each container by calculating the ratio of each container's CPU usage to the sum of all containers' CPU usage. Therefore, the power of a container e in a particular VM type can be estimated as follows,

$$P_{container_e} = {W_z}^T D + \frac{x_{cpu_e}}{\sum_r^h x_{cpu_r}} B, \tag{2}$$

where W_z is the weight vector of the z-*th* VM type corresponding to the VM in which the container e is located. D is the feature vector of the container corresponding to the feature vector of the VM. h is the number of containers on the server, and x_{cpu_r} represents the CPU usage of the r-*th* container.

This model realizes the power estimation of containers on VMs by cleverly using the VM model to estimate the power of the container to avoid the more massive fitting error stemming from twice modeling.

IV. Experiments

A. Experimental Setup

To simulate the environment of data centers, we built an experimental environment of OpenStack. There is an OpenStack controller node, an OpenStack compute node (Intel Core i5-6500 CPU, 2*4G DDR4@2400 MHz, 1TB HDD) and a monitor node. The monitor node collects the PDU data of the compute node through a serial line at a frequency of 1Hz (once per second). We deployed a collector called collectd [21] on the compute node, which can be used to collect OSM and PMC of the compute node, as well as to read these metrics of VMs through the hypervisor layer. The container's data are mainly obtained by executing the command `docker stats` of Docker (Docker is one kind of container). The sampling frequency of collectd and docker stats are also set to 1Hz.

For the features of collected data, we choose the most common counters in OSM, with CPU usage, memory usage, hard disk read/write bytes, and network interface receive/send bytes. Due to the contribution of hard disk read bytes and hard disk write bytes to power is the same [9], so we add

these two features together as one feature. We adopted OSM rather than PMC because counters in OSM are easy to collect on any system, even on containers running on the VMs. In contrast, counters in PMC (such as unhalted cycles) cannot be collected in such virtual environments. Since network IO tasks are not involved in PARSEC, this paper does not take network interface receive/send bytes as the model features in the experiments to simplify the model. In other tasks involving network IO, these two features should be taken into account.

The benchmark we used in our experiments is PARSEC v3.0 [22], as it is multi-threaded and diverse. This version of the suite contains 13 emerging programs that can be used to simulate CPU-intensive, memory-intensive, and IO-intensive workloads in the data center.

We deployed two VMs with two cores, 2G memory, and 50G hard disk on the compute node to run the benchmarks. For each VM, the workloads in PARSEC were run 50 times in a Docker container with a single thread and dual threads, respectively. Besides, we also ran a Docker container for the PARSEC with dual threads in each VMs at the same time to cover the situation where most of the resources were used, which represents 4-core load on the physical machine. We randomly divided the runtime data of each run into training sets (70%) and test sets (30%), and then spliced the training sets of all runs to obtain a whole training set of all workloads in PARSEC.

The evaluation metric we use in the experiments is Normalized Root Mean Square Error (NRMSE). The specific calculation equation is as follows,

$$NRMSE = \frac{\sqrt{\frac{1}{v}\sum_{i}^{v}\left(y_i - y_i'\right)^2}}{\sigma}, \tag{3}$$

where y_i represents the measured value, and y_i' represents the output value of the model. v is the number of samples, and σ is the standard deviation of all measurements.

B. Evaluation of VM Power Estimation

We first choose the least-squares linear model as a comparison, which is the most common power estimation model [8]–[10]. Secondly, based on observation, we segment the linear model with the case of {single-core full load, dual-core full load} as the split points, dividing the original linear model into the three-segment linear model. Consequently, the piecewise linear model is used as the second comparison model.

We choose the regression tree model in CART [23] as the third comparison model. Although the regression tree is selected as a comparison model, it does not have the capability of VM power estimation. Because its leaf node stores only a value, and the value cannot be directly used as the power estimation for a single VM, not to mention the power estimation of containers.

In the model we proposed, we assign the minimum subset size γ to 1% of the training set size. For the small training set, we stipulate that γ should not be less than 30. We assign the minimum error decrease threshold τ to 20.

We respectively tested the four models on the whole test set of all workloads and test sets of each workload. Finally, the comparison results of the models are shown in Table I.

TABLE I
PERFORMANCE (NRMSE) OF MODELS ON ALL WORKLOADS

Workload Name (Intensive Type)	Linear	Piecewise Linear	Regression Tree	Our Model
all workloads	0.771	0.768	0.453	**0.396**
blackscholes (CPU)	0.951	0.917	0.588	**0.541**
bodytrack (CPU)	0.887	0.865	0.725	**0.709**
canneal (CPU+Mem)	4.214	4.198	2.050	**1.272**
dedup (Mem+IO)	2.010	1.977	**1.295**	1.402
facesim (CPU)	0.919	0.916	0.378	**0.262**
ferret (CPU)	0.698	0.720	0.379	**0.331**
fluidanimate (CPU)	0.568	0.568	0.192	**0.165**
freqmine (CPU)	0.381	0.391	0.263	**0.231**
raytrace (CPU+Men)	**0.574**	0.587	0.858	0.602
streamcluster (CPU)	1.184	1.223	0.456	**0.348**
swaptions (CPU)	0.714	0.730	**0.180**	0.209
vips (CPU+IO)	1.454	1.433	0.876	**0.807**
x264 (CPU+IO)	1.169	1.152	0.650	**0.595**

On the test set of all workloads, our model's estimation error is smaller than the other three models, indicating that our model performs better than other models on most workloads. While the piecewise linear model has an improvement on the linear model, the improvement is not obvious, and it even drops a little on some workloads (such as ferret, freqmine, etc.). It reveals that the split points set manually based on core full-load states are not accurate, and the piecewise model may not fit the optimal power curve.

As shown in Table I, all models are not ideal under the workloads of canneal and dedup, and their NRMSE is above 1.0. Especially on the workloads of canneal, the NRMSE of the linear model and the piecewise model peaks above 4.0, which needs further study. We omit the box plot of all workloads' power due to space constraints, but we observed that the power ranges of the canneal and dedup are smaller than other workloads with the same number of processes. It is partly due to the type of workloads. Based on Table I, we can find that the workloads with excellent performance are mostly simple CPU-intensive workloads, and the dependencies between components that need to be represented by PMC are similar on these workloads, so they all have good performance. However, the dependencies between different components of memory-intensive and IO-intensive workloads are quite different from those of CPU-intensive workloads, resulting in large errors of power estimation. After comparing the power estimated by four models with the measured power, we found that the linear model is underfitted, and our model is more accurate than the linear model in most cases.

C. Evaluation of Container Power Estimation

With the VM power model established, we can estimate the container-level power. In Section III-B, we explained that the container can be regarded as a group of processes on the VM. The features of the container can be substituted

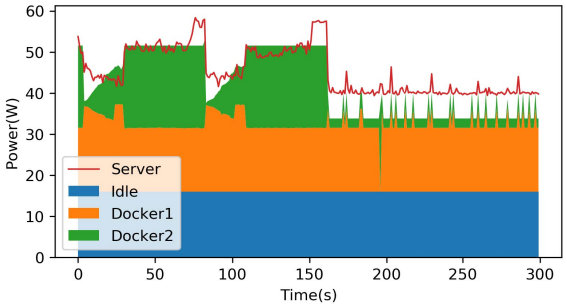

Fig. 2. The evaluation results of Dockers.

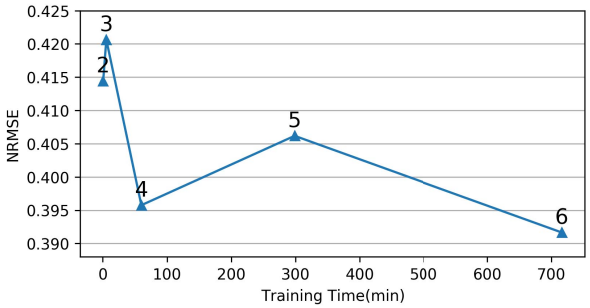

Fig. 3. Model performance and training time on different floating-point precision.

for the corresponding features of VMs in the VM model to estimate container power with the two prerequisites mentioned in Section III-B satisfied. Fig. 2 shows the evaluation results of containers on two VMs.

The line at the top of Fig. 2 is the power curve of the physical server. The bottom area represents the server's idle power, which is about 16W on the experimental machine. The middle two areas correspond to the power values of the Docker on VM 1 and the Docker on VM 2 respectively estimated by the VM model. To simplify the representation, we color the different workloads' Dockers in the same VM with the same color (such as workloads before and after 160s).

Although the two curves on the top of Fig. 2 are not entirely fit, the overall trends are the same. The main reason for the gap between Docker power and the physical machine power is when Dockers are running, one part of the VM power generated by the VMs in the process of scheduling Dockers is not reflected by Dockers' features. This part of power is not counted as the power generated by the containers in our model. Moreover, the gap is partly due to the power generated by the physical machine itself, such as the OpenStack's compute node survival detection, which is not included in the idle power.

We can also find that the sum of the idle power and the containers' power may exceed the physical machine's power. Although the power of the physical machine fluctuates slightly, it can be found that the container power of the interval [5,80] and interval [80,160] in Fig.2 is the same when Dockers are running the same workload. Moreover, the server's overall mean power is still relatively consistent with the sum of the container power. Therefore, the proposed model can estimate the power of the containers running on the VMs well.

D. Training Time and Floating-point Precision

In Section III-A, we mentioned that when selecting the best split points, we can reduce the time of traversing the split points by rounding the floating-point precision of data, to reduce the time of generating tree. Therefore, in this part, we experiment with the condition that the minimum subset size γ and the error decrease threshold τ are unchanged to reveal the influence of split point precision on the evaluation performance and training efficiency.

Fig. 3 is a figure of the relation between the tree generation time (X-axis) and the error of power estimation on the test set (Y-axis) with the total data volume of 316,590 when floating-point precision changes from 2 to 6. It can be seen that the overall trend of error is downward as floating-point precision increases while training times double. However, in some cases (such as when the precision is 3 or 5), the performance is worse than the previous one.

Although the overall trend of error is decreasing, we should consider whether it worth spending too much training time for little improvement. For example, it is probably not worth changing the precision from 4 to 6 to spend almost ten additional hours for a 1% decrease in NRMSE. Therefore, choosing 4 as the initial floating-point training precision is reliable with relatively high accuracy and acceptable speed. However, during the actual estimation, if the error exceeds a certain threshold due to the appearance of other task patterns not in the training set, 2 can be set as the floating-point precision to retrain the model. The new trained model can quickly reduce the power estimation error on tasks with other patterns by replacing the old model. Retraining the model with a short time alleviates the delay of scheduling requests.

V. Conclusion

In this paper, we propose a new container power estimation model by establishing a fast-training piecewise linear model between the VMs' features and the physical machine's power. The model combines good interpretability of linear regression and decision tree. Regarding a container as a process group of the VM, the container features are substituted into the VM model to realize the power estimation of the container on the VM. We test the performance of our model on 13 workloads of PARSEC and compare it with several models. Our model shows remarkable performance improvement on most workloads. Meanwhile, the power of the containers can be estimated as expected.

The major problem of our model is that the linear model does not reveal the dependencies between counters in OSM on different types of workloads. As a result, it cannot achieve outstanding performance on all workloads with a single training. Therefore, in future work, we will study the dependencies between these features to propose a more generalized power estimation model.

REFERENCES

[1] Z. Dong, W. Zhuang, and R. Rojas-Cessa, "Energy-aware scheduling schemes for cloud data centers on google trace data," in *IEEE Online Conference on Green Communications, OnlineGreenComm 2014, November 12-14, 2014*, 2014, pp. 1–6.

[2] S. F. Piraghaj, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "A Framework and Algorithm for Energy Efficient Container Consolidation in Cloud Data Centers," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*. IEEE, dec 2015, pp. 368–375.

[3] K. Kaur, T. Dhand, N. Kumar, and S. Zeadally, "Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers," *IEEE Wirel. Commun.*, vol. 24, no. 3, pp. 48–56, 2017.

[4] X. Yu, G. Zhang, Z. Li, W. Liangs, and G. Xie, "Toward Generalized Neural Model for VMs Power Consumption Estimation in Data Centers," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. IEEE, may 2019, pp. 1–7.

[5] W. Jiang, F. Liu, G. Tang, K. Wu, and H. Jin, "Virtual Machine Power Accounting with Shapley Value," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, K. Lee and L. Liu, Eds. IEEE, jun 2017, pp. 1683–1693.

[6] M. Colmant, M. Kurpicz, P. Felber, L. Huertas, R. Rouvoy, and A. Sobe, "Process-level power estimation in VM-based systems," in *Proceedings of the Tenth European Conference on Computer Systems - EuroSys '15*. New York, New York, USA: ACM Press, 2015, pp. 1–14.

[7] B. Krishnan, H. Amur, A. Gavrilovska, and K. Schwan, "VM power metering: feasibility and challenges," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 3, p. 56, jan 2011.

[8] M. Y. Lim, A. Porterfield, and R. Fowler, "SoftPower: fine-grain power estimations using performance counters," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC '10*. New York, New York, USA: ACM Press, 2010, p. 308.

[9] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proceedings of the 1st ACM symposium on Cloud computing - SoCC '10*. New York, New York, USA: ACM Press, 2010, p. 39.

[10] A. E. Husain Bohra and V. Chaudhary, "VMeter: Power modelling for virtualized clouds," in *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*. IEEE, apr 2010, pp. 1–8.

[11] R. Brondolin, T. Sardelli, and M. D. Santambrogio, "DEEP-Mon: Dynamic and Energy Efficient Power Monitoring for Container-Based Infrastructures," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, may 2018, pp. 676–684.

[12] G. Fieni, R. Rouvoy, and L. Seinturier, "Smartwatts: Self-calibrating software-defined power meter for containers," in *20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGRID 2020, Melbourne, Australia, May 11-14, 2020*, 2020, pp. 479–488.

[13] S. S. Tadesse, F. Malandrino, and C.-F. Chiasserini, "Energy Consumption Measurements in Docker," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2. IEEE, jul 2017, pp. 272–273.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[16] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer, "An energy efficiency feature survey of the intel haswell processor," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop, IPDPS 2015, Hyderabad, India, May 25-29, 2015*, 2015, pp. 896–904.

[17] V. Pallipadi, "Enhanced intel speedstep technology and demand-based switching on linux," *Intel Developer Service*, 2009.

[18] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*, 2010, pp. 1–8.

[19] C. Gu, S. Shi, P. Shi, H. Huang, and X. Jia, "Towards VM power metering: A decision tree method and evaluations," in *Algorithms and Architectures for Parallel Processing - 15th International Conference, ICA3PP 2015, Zhangjiajie, China, November 18-20, 2015, Proceedings, Part I*, 2015, pp. 508–523.

[20] B. Ibryam, "Principles of container-based application design," *Redhat Consulting Whitepaper*, 2017.

[21] F. Forster and S. Harl, "collectd – the system statistics collection daemon," 2012. [Online]. Available: https://collectd.org

[22] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," Princeton University, Tech. Rep. TR-811-08, January 2008.

[23] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.